Intergraph Modular GIS Environment (MGE)

MGE is a modular system that provides access to common utilities, applications software, and databases while accommodating a variety of workflows. The MGE modules facilitate the capture, storage, retrieval and analysis of geographic data. Responsible for creating, maintaining, and analyzing GIS/mapping databases, MGE is useful for both production and planning environments.

MGE stores geographic data in layers, allowing users to access data by selecting and querying on features of interest. The layered data model provides storage structures for the geometry and linkages to the relational database attribute records, while remaining almost transparent to the users. The layered implementation is accommodating to users that are familiar with manual mapping workflows and Computer-Aided Drafting (CAD) systems. MGE stores graphic data as MicroStation design (DGN) files; MGE can also link the graphics to attribute tables stored in Standard Query Language (SQL)-compliant relational Database Management Systems (DBMS) such as Oracle, Informix, or SQL Server.

The Spatial Data Standards/Facilities Management Standards (SDS/FMS) has built a geographic data model that allows spatial (graphic) and tabular (attribute) data to be related. As the SDS/FMS has been designed for applicability to existing GIS products, the SDS/FMS structure accommodates the MGE data model. As described later in this document, the SDS/FMS hierarchy and terminology seamlessly maps into the MGE structure. In order to create the MGE data structure for graphic and attribute data, the following three primary modules are needed: MGE Nucleus, MGE Basic Administrator, and MGE Base Mapper.

MGE Basic Nucleus (MGNUC)

MGNUC provides basic functions for project management and tools for data query, display, and review; it also provides projection coordinate systems.

MGE Basic Administrator (MGAD)

MGAD provides system and database management tools as well as setup and management routines for functionality in MGE Base Mapper. As a data modeling application, MGAD provides system management tools for creating and maintaining a data model.

MGE Base Mapper (MGMAP)

MGMAP contains modules for capturing, generating, cleaning, manipulating, and validating project data. MGMAP also contains a module for matching address descriptions to geographic positions.

Implementing the SDS/FMS using Intergraph MGE

Project Setup Overview

The following steps should be followed to establish the appropriate MGE working environment. Many of these steps will only need to be performed once when the project is set up, not for each individual session. The following chart shows the relationship between MGE Project Setup and SDS/FMS Implementation steps (see Figure 1).

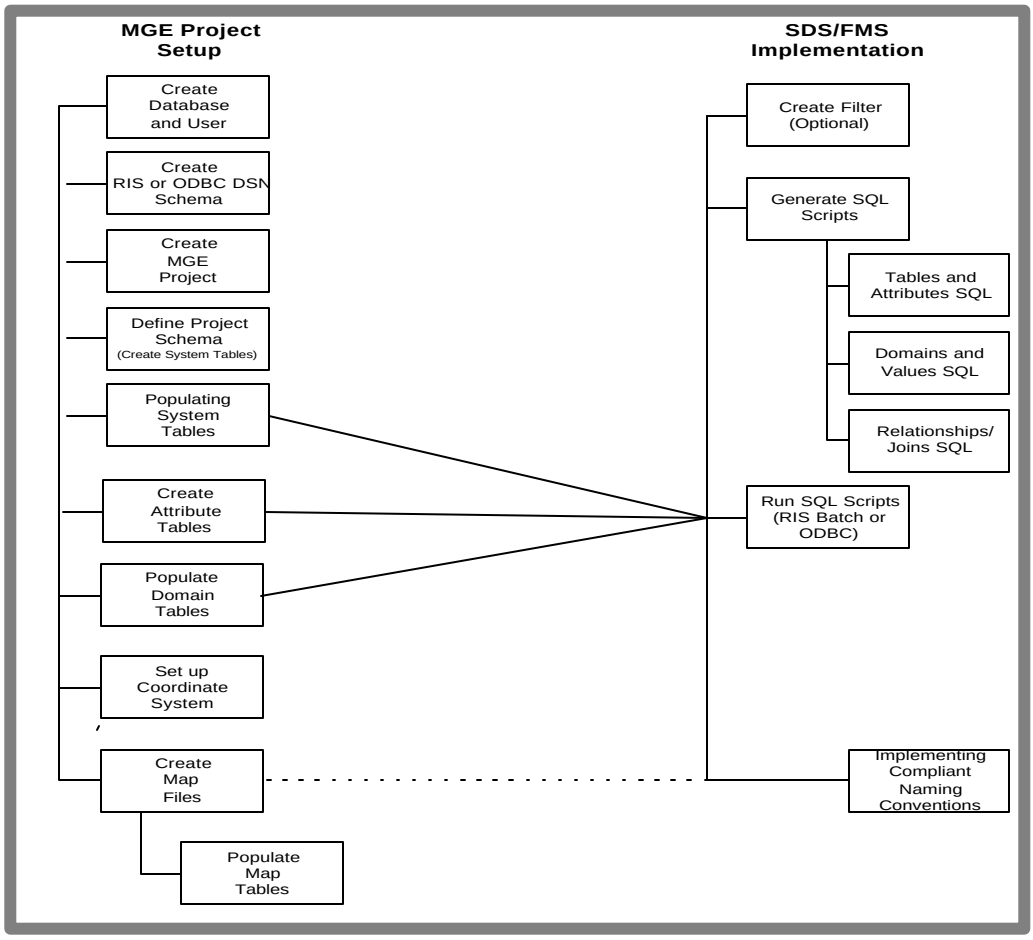

Figure 1. "MGE/SDS/FMS Implementation Diagram"

Creating the Database User and the Relational Interface System (RIS) Schema

In order to store attribute data that describes the geographic elements, a SQL-based relational Database Management System (DBMS) must be installed and configured. MGE can use various DBMSs because the attribute data in MGE is retrieved and manipulated the same way, regardless of the database used. Communication is established between the DBMS and MGE by RIS (Relational Interface System) or by ODBC drivers.

RIS and ODBC allow the users to access and manipulate the tables that are created and maintained by the database.

An MGE user's "view" on one of these user-specific table groups is called a schema. In order to build an MGE project, the schema must be defined prior to establishing the project. It is not mandatory to assign a schema to a project because MGE has a number of available functions for graphic data only. However, for implementing the SDS/FMS for tabular data, the schema must be defined.

The SDS/FMS does not specify how to set up user spaces in DBMSs or how to utilize the capabilities of the different DBMS systems. Oracle, Informix, and SQL Server are efficient in setting up client-server environments and managing large databases, and both can be used with MGE RIS or ODBC. The RIS client is resident in the MGE software, but the RIS Server must be purchased separately. The ODBC Drivers are usually included with the DBMS.

Creating the MGE Project and Defining Project Schema

To organize the data efficiently, all information to be processed by MGE is held and combined in a Project. When creating a Project, MGE creates a project file, assigns a RIS/ODBC schema, and creates directories where graphic and auxiliary data will be stored.

Directory and File Naming

The SDS/FMS does not specify how to organize the graphic data when the same feature is presented on multiple maps. It is advisable that the user establishes a directory structure to help organize large data sets. Only the design file names that are defined for SDS/FMS Entity Classes are applicable in MGE. However, each dataset may require identification in a project-specific directory structure; and, therefore, the SDS/FMS map names will have to be modified. For example, if the maps are organized by counties, each county will have a "trveh.dgn," which will not identify the files uniquely and which may cause problems when transferring files under the same name. By creating a directory for each county, the files may be differentiated in this example.

Table Relationships between SDS/FMS and MGE

The following table (Figure 2) shows the relationships between SDS/FMS and MGE.

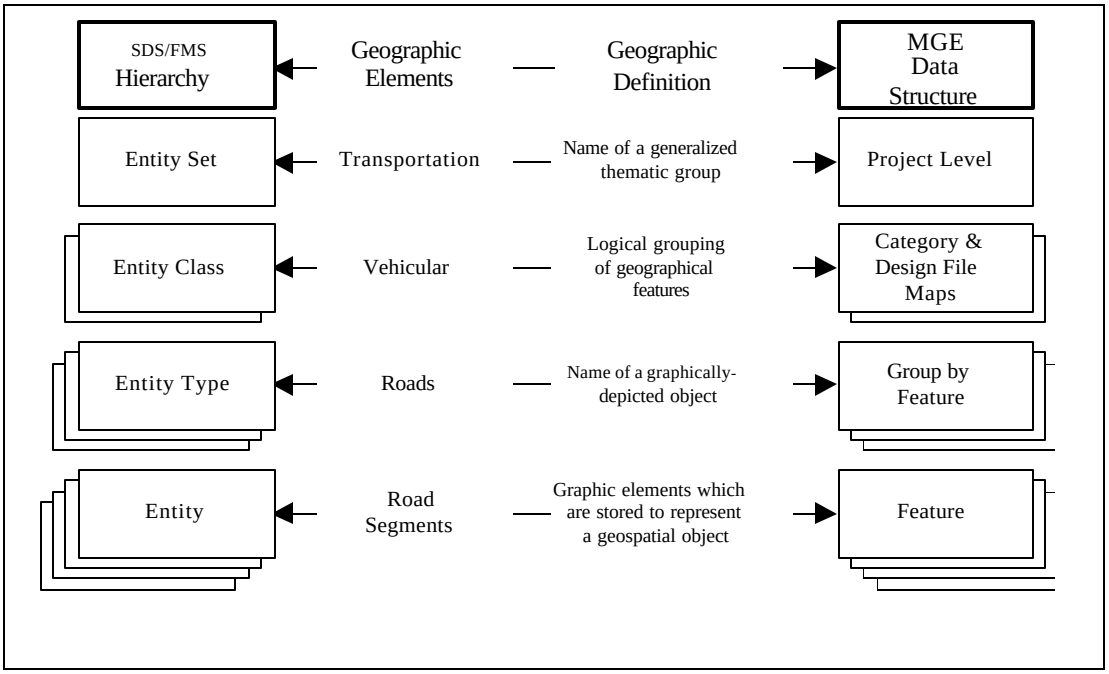

Figure 2. "SDS/FMS/MGE Nomenclature Comparison"

The following terms are used by MGE to group and organize spatial and non-spatial data relevant to a task:

Project

The MGE project is the combination of geographic information (geographic objects and attribute data) from all sources.

The project can be organized into indexes, which are design files containing index shapes that group features of categories into geographic themes. The indexes correlate to the Entity Class in the SDS/FMS. Note: Setting up indexes and index files is optional; i.e., the category table created by the SDS/FMS scripts does not have index values.

Maps

Geographic information in MGE is stored in the system as maps. These maps represent graphic information and relate to Entity Classes in the SDS/FMS. MGE stores graphic information in MicroStation Design Files (.dgn)

Categories

Categories, within MGE, represent a group of thematically or geographically related features or maps. In SDS/FMS, each category can have one or more maps.

Features

Features are spatially distributed geographic elements that make up a map. A geographic element in the SDS/FMS relates to an Entity Type or Entity. By the SDS/FMS definition,

these elements are represented on the map as points, lines, polygons, text and attribute information, which are all individual features, and may be associated with an attribute table. These elements are classified in the SDS/FMS as entities.

Attributes

Attributes are non-graphical information describing features and files or relating to features and files. Attributes are stored in attribute tables.

Populating System Tables

The MGE data integrity is secured and the data model is established by setting up system tables for the storage of graphic data characteristics and the attribute data structure. In the MGE system tables, the primary and foreign keys are referred to as identifier (mslink) and join columns. These columns are populated automatically by running the SDS/FMS SQL script created by the SDS/FMS Generator and/or by following the MGE procedures explained in the following sections.

After creating the project, the "Define Project Schema" must be used to create the following system tables:

Mscatalog, Feature, Category, Maps, Attribute_Catalog, Domain_Catalog, Range_Domain, List_Domain, View_Content, View_Catalog, View_Join, Join_Catalog, and Label.

The following table (Figure 3) contains terms that are used by MGE to group and organize spatial and non-spatial data relevant to a task:

Tablename	Description	Input Data Provided/Defined by the SDS/FMS	Data Populating Method
Mscatalog	Specific to MGE/MicroStation	-	Populated by MGE
Feature	Stores the names of geographic elements, their symbology and characteristics	SQL code generated by the tool Tables and Attributes within the SDS/FMS Generator	-
Category	Defines the groups of geographic elements	SQL code generated by the tool Tables and Attributes within the SDS/FMS Generator	-
Maps	Stores the name of maps in which the geographic elements are stored	-	Populated by MGE by using map loader
Attribute_Catalog	Stores the names and field names of attribute tables	SQL code generated by the tool Tables and Attributes within the SDS/FMS Generator	-
Domain_Catalog	Creates the link between the attribute tables and the domain values	SQL code generated by the tool Domains and Values within the SDS/FMS Generator	-
Range_Domain	Stores range domain values	-	-
List_Domain	Stores list domain values	SQL code generated by the tool Domains and Values within the SDS/FMS Generator	-
View_Content	-	-	Populated by user in MGE
View_Catalog	-	-	Populated by user in MGE
View_Join	-	-	Populated by user in MGE
Join_Catalog	-	SQL code generated by the tool Relationships/Joins within the SDS/FMS Generator	-
Label	-	-	Populated by user in MGE

Note: Some of the entries shown in the tables in figure X are optional. To utilize all the MGE functions, all of the above tables and links must be created.

Figure 3. "Table Relationships between SDS/FMS and MGE"

Populating Category, Feature, Mscatalog, and Attribute_Catalog Tables

The SDS/FMS code generated by the Tables and Attributes tool, populates the Category, Feature, Mscatalog, and Attribute_Catalog tables with the necessary records. All rows include mslink, category name, and index type. A SQL script can be generated for the entire SDS/FMS or the user can generate a SQL script based on one of six predefined filters or a user created filter. These filters are used to create a subset of SDS/FMS by creating a smaller database.

The user can also use the Filter Maker tool now available to create a custom filter for use within the SDS/FMS. This gives the user the ability to choose only those entities or features that are necessary for their project.

In order to create a SQL script for loading the tables and attributes into the database, the following options are currently available.

1. Open the SDS/FMS Generator. Under the Generator/New pulldown, select the tool Tables and Attributes.
2. Select the required Entity Sets and Entity Classes or the required Filter from the SDS/FMS Generator Interface.
3. Generate the SQL script for all the selected Entity Sets or Filter. The SQL script is saved as an ASCII file in a directory and name of your choosing.
4. Run the script in RIS Batch. This will load the script directly into the database or ODBC method.

Populating Domain Tables

For management of domain values, the SDS/FMS generator creates a script to populate the domain_catalog table and creates the necessary domain tables. This script can contain all the domain values within the SDS/FMS or it can contain just those domains that are associated with a particular Filter.

In order to create a SQL script for the purpose of loading domain values into the database, the following options are available.

1. Open the SDS/FMS Generator. Under the Generator/New pulldown, select the tool Domains and Values.
2. Select the required option (All Domains, Single Domains, or Prior Generation). All Domains will generate a SQL script containing all the Domains within the SDS/FMS, Single Domain will generate a SQL script with a single Domain, and Prior Generation will create a SQL script containing those Domains associated with a Filter or the file from the Tables/Attributes generation.
3. Generate the SQL script for all the selected Domains. The SQL script is saved as an ASCII file in a directory and name of your choosing.
4. Run the script in RIS Batch. This will load the script directly into the database or ODBC method.

Populate Join Catalog

The SDS/FMS generator creates a script to populate the join catalog table. This will create links between attribute tables so that all the existing and relevant information of features can be accessed and analyzed in MGE or the DBMS.

In order to create a SQL script for the purpose of loading joins into the database, the following options are available.

1. Open the SDS/FMS Generator. Under the Generator/New pulldown, select the tool Relationships/Joins.
2. Select the required option (All Joins, Single Joins, or Prior Generation). All Joins will generate a SQL script containing all the Joins within the SDS/FMS, Single Join will generate a SQL script with a single Join, and Prior Generation will create a SQL script containing those Joins associated with a Filter.
3. Generate the SQL script for all the selected Joins. The SQL script is saved as an ASCII file in a directory and name of your choosing.

4. Run the script in RIS Batch. This will load the script directly into the database or ODBC method.

See the section (An Electrical Implementation) for a step-by-step approach to creating filters and scripts.

Create Views

To take full advantage of the Joins, a view or views may need to be created within MGE to fully exploit the underlying attribute database and to associate relating tables and features. The views that are needed are generally specific to the project structure and the data model; the SDS/FMS does not restrict or enforce creating views in any way.

Create Map Files and Setup Coordinate System

When creating new graphic data, files need to be created in the MGE environment. A primary and secondary coordinate system can be defined, and the data can be easily transformed to other projection systems.

Populate Map Tables/Establishing Map Naming Conventions

The *Maps Table* in MGE contains information for maps/design files. This information includes the map name and category name to which the map belongs. The Maps Table is created by Define Project Schema and is populated by "Map Manager."

A SDS/FMS-compliant naming convention should be established for the user data set, to meet the following requirements:

- Uniquely identify files.
- Classify files into Entity Sets and Entity Classes based upon the naming convention.
- Develop a directory structure/hierarchy of maps/design files.

SDS/FMS naming conventions define unique names for each MGE map, corresponding to Entity Classes. The DOS standard file name allows an 8-character description and a 3-character extension. Within the SDS/FMS structure, 5 characters are used to identify and categorize the file. Within the 5 characters, the first 2 identify the Entity Set of the file and the next 3 identify the file's Entity Class. The remaining 3 characters within the filename allow for further identification.

Depending on the number of maps and naming criteria, more than the above mentioned 3 characters may be needed. The character identification for Entity Set and Class may be consolidated into 3 or 4 characters and the 3 character extension may also be different, e.g., the state IDs, county or town name abbreviations, facility IDs, and building numbers can be efficiently included and "decoded" by establishing a unique naming convention.

The SDS/FMS does not describe the directory structure to further specify the graphic data storage hierarchy. MGE creates the .dgn directory under the project directory. However, any file can be opened and processed by MGE from other directories.

Populating Metadata Tables

Providing metadata on the graphics and

attribute tables is essential to documenting the validity of datasets. For instance, the SDS/FMS can be used to meet Federal Geographic Data Committee (FGDC) requirements for metadata in support of the National Spatial Data Infrastructure (NSDI). The SDS/FMS dataset generates a code to create and populate the tables with descriptive information on the underlying data structure and the methodology of describing/capturing the "real world" objects.

In the MGE environment, there is no user-friendly tool to view metadata directly, as the main functions are built primarily to manipulate feature-attribute data. However, in the Query Interface, any table registered in the attribute_catalog can be queried and rows can be retrieved, including metadata. To populate the metadata tables, SQL +, SQL Editor, or RIS Interactive can be used.

SDS/FMS Implementation Scenario

The following scenario illustrates the process when planning a new project for the application of the SDS/FMS standards, starting with geographic features and attribute information. Both the MGE structure and the SDS/FMS must be thoroughly understood for full compliance.

Implementation for Graphic and Attribute Data

Figure 1 describes how the SDS/FMS can be used to implement the Entity and Attribute table structure when setting up an MGE project.

First, a database user and an empty database have to be created. Second, a RIS Schema or ODBC DSN has to be created. Third, a MGE project has to be created.

The user should now generate the three SQL scripts (Tables and Values, Domains and Values, and Relationships/Joins) that will be used to populate the MGE tables and to create the necessary attribute tables. In turn these scripts populate the Category, Feature, Mscatalog, Attribute_Catalog, Domain_Catalog, and Join_Catalog tables. By generating the script files with an active Filter, only those Entities necessary for a particular project will be loaded.

The SQL script file should now be loaded into the previously empty database. RIS Batch may be used if RIS is the link to the database. Any SQL batch routine may be used if ODBC is the chosen link.

The data structure is now established to enter, update, and manipulate new attribute data. If there is existing attribute data for the project, it has to be loaded into the previously created attribute tables.

For graphic data, the maps table has to initially be populated. For complying with the SDS/FMS naming conventions during this process, refer to the section "Populating Map Tables/ Establishing Map Naming Conventions." MGE has a Map Loader and Map Manager tool to accomplish these steps in a user-friendly environment.

Finally, the features can be digitized in the design files/maps or electronically integrated if already in digital form. Using MGE Base

Mapper, feature conversion of design file elements can be performed or the features can be digitized by selecting the feature names and by the embedded digitizing commands specific to features. The Define Attribution and the Feature/Attribute Manager tools of the MGE Base Mapper provide a user-friendly interface to enter the attribute values by features.

A number of other tools, such as SQL+, SQL Editor, RIS Interactive, and the MGE Base Mapper's Bulk Update can also be used to update fields in the database (See manuals for selected RDBM's for SQL specific key-ins, RIS Utilities Guide, and MGE Base Mapper Users Guide.

By using the linework cleaning tools, the graphics can be checked and corrected to prepare the maps for further processing (e.g., creating topology.) Area and perimeter values can also be loaded into the attribute tables.

By careful execution of the above steps, the data is now available for further analysis. The SDS/FMS efficiently facilitates the maintenance and GIS processing of the data, as it has already been tested in a number of implementations.

An Example Electrical Implementation

Following is a sample implementation using an exterior lighting data set as an example:

1. A user and an empty database must be created using the DBMS of choice to hold the electrical information.

2. If RIS is the link to the electrical database, a RIS Schema must be created. This is done using the RIS Schema Manager routine.

   If ODBC is the link to the electrical database, a DSN must be created using the ODBC administrator.

   The RIS Schema or ODBC DSN should be something descriptive, i.e., ELECTRICAL.

3. A MGE project must now be created. MGE should be selected from the START menu and the FILE/NEW option selected from the pull down. The user should key in the new project name (i.e., ELECTRICAL) and select a home directory for the new project. If the directory does not exist, MGE will create it.

   The user should then select the appropriate RIS Schema or ODBC DSN. This will associate the database with the MGE project. MGE will create the project and associate the database with it.

4. The database will be populated with the MGE required system tables. Selecting the Define Project Schema tool found in the MGE Basic Administrator does this. The Defined Project Schema will open with the correct Schema name and a list of tables. The user should select OK. MGE will generate the necessary tables in the electrical database.

5. A filter should now be created using the SDS/FMS Filter Maker, as follows:

a. Select the SDS/FMS Filter Maker and connect to the correct SDS/FMS standard. If it is already connected select NEXT.

b. Key in the desired name for the Filter you want to create and select NEXT.

c. Locate the desired directory and name for the SDF file and select NEXT.
d. The dialog box will build a list of features within the SDS/FMS. When this list is complete, scroll down and select all the electrical and exterior lighting features. Then, select NEXT.

e. Select the CREATE button and the Generator will create a user defined filter containing only the electrical and exterior lighting features.

6. The SQL Script files should now be created using the SDS/FMS Generator and the Filter created in Step 5.

a. Select the SDS/FMS Generator from the START menu.

b. Select the Tables and Attributes tool from the Generate/New pulldown menu.

c. Select the Custom Feature Schema option and then select the Filter created in Step 5.

d. In the Generate portion of the dialog box, select the SQLCode.SQL with a double click of the mouse. This will open a dialog box that allows the user the select the directory and name of the SQL file. For this purpose we could name it ELECTRICAL.SQL. After the name is keyed in, select SAVE.

e. In the Default Schema portion of the Generate dialog box key in the name of the schema created in Step 2. In this case, the name would be ELECTRICAL.

f. The GENERATE button should now be selected to create the first SQL script file.
g. Select the Domains and Values tool from the Generate/New pulldown menu

h. Select the Prior Generation option from the dialog box; then select Locate Generation SQL. A dialog box will open allowing the user to select the SQL file created in Step 6F. This SQL file contains those features that are associated with the filter created in Step 5.
Once the SQL file has been identified, select OPEN. A list of Domains associated with this filter will appear in the left-hand section of the dialog box.

i. In the Generate portion of the dialog box, select the SQLCode.SQL with a double click of the mouse. This will open a dialog box that allows the user the select the directory and name of the SQL file that will contain the domains. For this example, we could name it ELE_Domain.SQL. After the name is keyed in, select SAVE.

j. In the Default Schema portion of the Generate dialog box, key in the name of

the schema created in Step 2. In this case the name would be ELECTRICAL.

k. The GENERATE button should now be selected to create the second SQL script file.

l. Select the Relationships/Joins tool from the Generate/New pulldown menu.

h. Select the Prior Generation option from the dialog box; then select Locate Generation SQL. A dialog box will open allowing the user to select the SQL file created in Step 6F. This SQL file contains those features that are associated with the filter created in Step 5.
Once the SQL file has been identified, select OPEN. A list of tables associated with this filter will appear in the left-hand section of the dialog box.

i. In the Generate portion of the dialog box, select the SQLCode.SQL with a double click of the mouse. This will open a dialog box that allows the user to select the directory and name of the SQL file that will contain the joins. For this purpose we could name it ELE_Join.SQL. After the name is keyed in, select SAVE.

j. In the Default Schema portion of the Generate dialog box, key in the name of the schema created in Step 2. The sample name would be ELECTRICAL.

k. The GENERATE button should now be selected to create the third SQL script file.

7. Now that the three SQL files have been created, it is time to load the database with the SQL files.

   a. If you are using RIS, select the program RIS Batch from the Start/Programs menu. This will open a batch window. Key in the READ and drag the first SQL file (ELECTRICAL.SQL) into the window and drop it. Make the window active with a data point and hit return. The RIS Batch program will read the script file and load the information to the database.

   If you are using ODBC, select the appropriate SQL tool (i.e. SQL+) and key in the database user and password. When the window opens, drag the first script file into the window and drop it. Hit the execute button. The SQL routine will load the script file into the database.

   b. The same procedure talked about in 7A should be used to load the second and third SQL file (ELE_Domain.SQL) and Join (ELE_Join.SQL).

8. Now that the database has been loaded with the electrical structure (features, tables, etc.) it is time to begin working with the graphic files. The graphic files containing the electrical graphic data should move into the MGE project directory and be placed in the DGN the directory. All MGE routines can now be

used against the design file. Features can be created, tables loaded, etc.